International Journal of Mathematics and
Computer Applications Research (IJMCAR)
ISSN(P): 2249-6955; ISSN(E): 2249-8060
Vol. 5, Issue 2, Apr 2015, 39-52
© TJPRC Pvt. Ltd.

TWO PARALLEL NUMERICAL ALGORITHMS FOR SOLVING TRIDIAGONAL SYSTEM OF LINEAR EQUATIONS USING PARALLEL SPLITTING GAUSS – JORDAN AND THE GENERAL PARALLEL SPLITTING GAUSS – JORDAN METHODS

OSAMA EL-GIAR

Department of Basic Science, Modern Academy for Engineering and Technology in Maadi, Cairo, Egypt

ABSTRACT

In this paper we introduce the design and implementation of two parallel algorithms for solving the general tridiagonal linear system of equations using two new parallel algorithms: Parallel splitting Gauss – Jordan and the general parallel splitting Gauss Jordan algorithms. These algorithms are very easily distributed over different number of processors. In addition our implementations are very efficient, generally a linear speed-up is obtained by increasing the number of processors.

KEYWORDS: Parallel Algorithms, Numerical Analysis, Tridiagonal Linear System of Equations, Gauss Jordan Algorithm

1. INTRODUCTION

During the past few years, there has been an increasing number of parallel algorithms using many kinds of parallel computers to solve very large scientific problems. The main problems for designing parallel algorithms are:

1. How to break up the problem into smaller subproblems to be treated independently.
2. How to reorder the problem by restricting the sequence of operations to increase the amount of parallelism.

In this paper we consider the design and implementation of two new parallel algorithms for solving tridiagonal systems of equations arising from the numerical solution of partial differential equation using finite difference approximation.

Direct techniques: Gauss-Jordan method, WZ factorization and QR algorithms are particularly suited to parallel computers. In the past few years, there were some triers to solve the tridiagonal systems in parallel, see for example Hatzopoulos [2], who made the DWZ algorithm, but there was a restriction on $n = 2^m, m = 1, 2, \ldots$ and Henk, see [3] who made some alternative decomposition for the tridiagonal systems of equations. In this paper we will discuss the sequential Gauss-Jordan algorithm and the parallel version of it done by Evans, (see [1]) to solve a general tridiagonal system, of equations and we will introduce two new parallel algorithms and show their advantages.

2. SEQUENTIAL GAUSS-JORDAN ALGORITHM

Consider the tridiagonal system of equation $Ax = b$ (1)

Where A is an $n \times n$ tridiagonal matrix of coefficients, x and b are vectors of order n of unknowns and right hand side respectively given by

$$\begin{bmatrix} b1 & c1 & & & & \\ a2 & b2 & c2 & & & \\ & . & . & . & & \\ & & . & . & . & \\ & & & . & . & . \\ & & & & b_n & c_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ . \\ . \\ . \\ d_n \end{bmatrix} \tag{2}$$

The sequential algorithm is given by:

$$g_1 = \frac{c_1}{b_1}, \quad h_1 = \frac{d_1}{b_1}$$

$$\begin{cases} for \quad i = 2, 3, \dots, n \\ g_i = \dfrac{c_i}{b_i - a_i g_{i-1}} \;, \; i \neq n \;, \quad h_i = \dfrac{d_i - a_i h_{i-1}}{b_i - a_i g_{i-1}} \end{cases} \tag{3}$$

and the solution can be obtained by $x_n = h_n, \;\; x_i = h_i - g_i x_{i+1}, \;\; i = n-1, \; n-2, \dots, 1$

In the sequential Gauss-Jordan algorithm the steps are almost sequential and we can only calculate each g_i and h_i in parallel see figure 1 for the sequential Gauss-Jordan with n = 5.

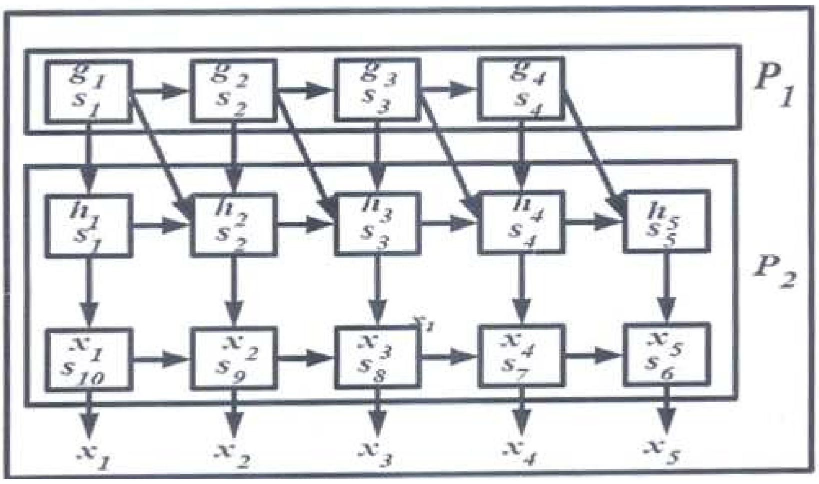

Figure 1: Sequential Gauss-Jordan for *n* = 5

Note that we need 10 sequential level for n = 5, i.e. in general we need 2n sequential operations for $n \times n$ system to be solved.

3. PARALLEL GAUSS-JORDAN ALGORITHM

In this section we will discuss the parallel Gauss-Jordan algorithm done by [1, J. V. Evans], the idea of this algorithm is that, after normalization of each element in the main diagonal, the elimination process eliminates coefficients both below and above it at the same time, until the original tridiagonal system is completely decoupled and has been transformed into the unit matrix and the solution is immediately available. The parallel Gauss- Jordan can be defined as

$$g_1 = \frac{c_1}{b_1}, \quad h_{1,0} = \frac{d_1}{b_1}$$

$$\begin{cases} for \quad i = 2, 3, \dots, n \\ g_i = \dfrac{c_i}{b_i - a_i g_{i-1}}, i \neq n, \quad h_{i,0} = \dfrac{d_i - a_i h_{i-1,0}}{b_i - a_i g_{i-1}} \\ \begin{cases} for \ k = i-1, i-2, \dots, 1 \\ h_{k,i-k} = h_{k,i-k-1} + (-1)^{i-k} \prod_{j=k}^{i-1} g_i h_{i,o} \end{cases} \end{cases} \tag{4}$$

and the solution can be obtained by $x_i = h_{i,n-i}, \ i = n, n-1, \dots, 1$.

The amount of parallelism is now increased and we need 5 sequential levels, for n = 5 and each level can be done in parallel see figure 2, the computations proceeds as follows:

<u>for sequential level 1 (*i* = *1*)</u>

Compute in parallel, g_1 and $h_{1,0}$.

<u>for sequential level 2 (*i* = *2*)</u>

Compute in parallel, g_2 and $h_{2,0}$ and $h_{1,1} = h_{1,0} - g_1 h_{2,0}$.

<u>for sequential level 3 (*i* = *3*)</u>

Compute in parallel, g_3, $h_{3,0}$ and

$h_{2,1} = h_{2,0} - g_2 h_{3,0}$

$h_{1,2} = h_{1,1} + g_1 g_2 h_{3,0}$

<u>for sequential level 4 (*i* = *4*)</u>

Compute in parallel, g_4, $h_{4,0}$ and

$h_{3,1} = h_{3,0} - g_3 h_{4,0}$

$h_{2,2} = h_{2,1} + g_2 g_3 h_{4,0}$

$h_{1,3} = h_{1,2} - g_1 g_2 g_3 h_{4,0}$

<u>for sequential level 5 (*i* = *5*)</u>

Compute in parallel, g_5, $h_{5,0}$ and

$h_{4,1} = h_{4,0} - g_4 h_{5,0}$

$h_{3,2} = h_{3,1} + g_3 g_4 h_{5,0}$

$h_{2,3} = h_{2,2} - g_2 g_3 g_4 h_{5,0}$

$h_{1,4} = h_{1,3} + g_1 g_2 g_3 g_4 h_{5,0}$

and the solution can be obtained in parallel at the same sequential step using

4. THE PARALLEL SPLITTING GAUSS-JORDAN ALGORITHM

The idea of the parallel splitting Gauss-Jordan algorithm is that the n×n system is splitted into two independent $\frac{n}{2}\times\frac{n}{2}$ systems and then each system can be treated independently and in parallel, but in this case we treat the first $\frac{n}{2}$ equations using the Gauss-Jordan algorithm 3 starting from i = 0 to i = $\frac{n}{2}$ and treat the second system at the same time using the backward Gauss-Jordan algorithm 9 starting from i = n backward to i=$\frac{n}{2}+1$.

Now to split the original system 2 into two independent $\frac{n}{2}$ systems of equations, let n be even, and we have to get red from $c_{\frac{n}{2}}$ in the $\frac{n}{2}^{th}$ equation and get red from $a_{\frac{n}{2}+1}$ in the $\left(\frac{n}{2}+1\right)^{th.}$ equation thus let: $\beta x_{\frac{n}{2}} = x_{\frac{n}{2}} + 1$ (5)

hence $x_{\frac{n}{2}+1}$ and $x_{\frac{n}{2}}$ can be eliminated from the $\left(\frac{n}{2}\right)^{th.}$ equation and the $\left(\frac{n}{2}+1\right)^{th.}$ respectively and the two new systems will be

$$\begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & & & & a_{\frac{n}{2}} & \left(b_{\frac{n}{2}} + \beta c_{\frac{n}{2}}\right) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_{\frac{n}{2}} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ \cdot \\ d_{\frac{n}{2}} \end{bmatrix} \quad (6)$$

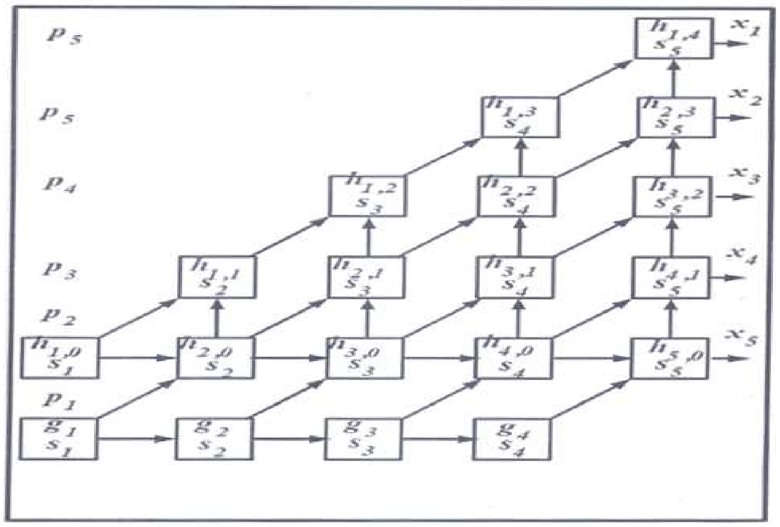

Figure 2: Parallel Gauss Jordan for n = 5

which will be treated by Gauss-Jordan algorithm 2 but for $i = 1, 2, ..., \frac{n}{2}$ and in this case

$$x_{\frac{n}{2}} = h_{\frac{n}{2}} = \frac{d_{\frac{n}{2}} - a_{\frac{n}{2}} h_{\frac{n}{2}-1}}{\left(b_{\frac{n}{2}} + \beta c_{\frac{n}{2}}\right) - a_{\frac{n}{2}} g_{\frac{n}{2}-1}} \tag{7}$$

and the second system will be

$$\begin{bmatrix} (b_{\frac{n}{2}+1} + \frac{a_{\frac{n}{2}+1}}{\beta}) & c_{\frac{n}{2}+1} & & & \\ a_{\frac{n}{2}+2} & b_{\frac{n}{2}+2} & c_{\frac{n}{2}+2} & & \\ & . & . & . & \\ & & . & . & . \\ & & & . & . \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_{\frac{n}{2}+1} \\ x_{\frac{n}{2}+2} \\ . \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} d_{\frac{n}{2}+1} \\ d_{\frac{n}{2}+2} \\ . \\ . \\ . \\ d_n \end{bmatrix} \tag{8}$$

which will be treated using the backward gauss-Jordan algorithm defined by:

$$g_n = \frac{a_n}{b_n}, \qquad h_n = \frac{d_n}{b_n}$$
$$\begin{cases} for \quad i = n-1, n-2, ..., \frac{n}{2}+1 \\ g_i = \frac{a_i}{b_i - c_i g_{i+1}}, \quad i \neq \frac{n}{2}+1, \quad h_i = \frac{d_i - c_i h_{i+1}}{b_i - c_i g_{i+1}} \end{cases} \tag{9}$$

and the solution can be obtained by $x_{\frac{n}{2}+1} = h_{\frac{n}{2}+1}$, $x_i = h_i - g_i x_{i-1}$, $i = \frac{n}{2}+2, \frac{n}{2}+3, \ldots, n$ and in this case

$$x_{\frac{n}{2}+1} = h_{\frac{n}{2}+1} = \frac{d_{\frac{n}{2}+1} - c_{\frac{n}{2}+1} h_{\frac{n}{2}+2}}{(b_{\frac{n}{2}+1} + \frac{a_{\frac{n}{2}+1}}{\beta}) - c_{\frac{n}{2}+1} g_{\frac{n}{2}+2}} \tag{10}$$

thus from equation 5, equation 7 and equation 10 we get

$$\beta = \frac{(b_{\frac{n}{2}} - a_{\frac{n}{2}} g_{\frac{n}{2}-1})(d_{\frac{n}{2}+1} - c_{\frac{n}{2}+1} h_{\frac{n}{2}+2}) - a_{\frac{n}{2}+1}(d_{\frac{n}{2}} - a_{\frac{n}{2}} h_{\frac{n}{2}-1})}{(b_{\frac{n}{2}+1} - c_{\frac{n}{2}+1} g_{\frac{n}{2}+2})(d_{\frac{n}{2}} - a_{\frac{n}{2}} h_{\frac{n}{2}-1}) - c_{\frac{n}{2}}(d_{\frac{n}{2}+1} - c_{\frac{n}{2}+1} h_{\frac{n}{2}+2})} \tag{11}$$

For example let n = 8 then the computation proceeds as follows:

<u>for sequential level ($i = 1$)</u>

In parallel compute: g_1, h_1 using Gauss-Jordan algorithm and g_8, h_8 using backward Gauss-Jordan algorithm 9.

for sequential level 2 (*i = 2*)

In parallel compute: g_2, h_2 using Gauss-Jordan algorithm and g_7, h_7 using backward Gauss-Jordan algorithm 9.

for sequential level 3 (*i = 3*)

In parallel compute: g_3, h_3 using Gauss-Jordan algorithm and g_6, h_6 using backward Gauss-Jordan algorithm 9.

for sequential level 4 (*i = 4*)

In parallel compute: β using equation 11

for sequential level 5 (*i = 5*)

In parallel compute: $h_4 = x_4$, $h_5 = x_5$ using equation 7 and equation 5 respectively

for sequential level 6 (*i = 6*)

In parallel compute: x_3 and x_6 using Gauss-Jordan algorithm and backward Gauss-Jordan algorithm 9.

for sequential level 7 (*i = 7*)

In parallel compute: x_2 and x_7 using Gauss-Jordan algorithm and backward Gauss-Jordan algorithm 9.

for sequential level 8 (*i = 8*)

In parallel compute: x_1 and x_8 using Gauss-Jordan algorithm and backward Gauss-Jordan algorithm 9. So in this case the amount of parallelism has been increased and we need only n sequential steps for n unknowns. see figure 3 for the Parallel splitting Gauss-Jordan with n = 8.

5. THE GENERAL PARALLEL SPLITTING GAUSS-JORDAN ALGORITHM

The idea of the general parallel splitting Gauss-Jordan algorithm is that the $n \times n$ system is splitted into two independent $\frac{n}{2} \times \frac{n}{2}$ systems and then each system can be treated independently and in parallel, and at the same time the first system is using the parallel Guass-Jordan algorithm 4 (from $i = 1,2, \dots, \frac{n}{2}$) and the second system is using the parallel backward Gauss-Jordan algorithm starting from i=n backward to i=$\frac{n}{2}+1$ defined by:

$$g_n = \frac{a_n}{b_n}, \qquad h_{n,0} = \frac{d_n}{b_n}$$

$$\begin{cases} for \quad i = n-1, \ n-2, \ \dots, \ \frac{n}{2}+1 \\ g_i = \dfrac{a_i}{b_i - c_i\, g_{i+1}}, \quad i \neq \frac{n}{2}+1, \quad h_{i,o} = \dfrac{d_i - c_i h_{i+1,0}}{b_i - c_i g_{i+1}} \\ \quad \begin{cases} for \quad k = i+1, \ i+2, \ \dots, \ n \\ h_{k,k-i} = h_{k,k-i-1} + (-1)^{k-i} \prod_{j=i+1}^{k} g_j \, h_{i,0} \end{cases} \end{cases} \tag{12}$$

and the solution can be obtained by $x_i = h_{i,i-\left(\frac{n}{2}+1\right)}, \quad i = \frac{n}{2}+1, \frac{n}{2}+2, \dots, n$

and similar to the parallel splitting Gauss-Jordan algorithm, β can be obtained by

$$\beta = \frac{(b_{\frac{n}{2}} - a_{\frac{n}{2}}\, g_{\frac{n}{2}-1})(d_{\frac{n}{2}+1} - c_{\frac{n}{2}+1}\, h_{\frac{n}{2}+2,0}) - a_{\frac{n}{2}+1}(d_{\frac{n}{2}} - a_{\frac{n}{2}}\, h_{\frac{n}{2}-1,0})}{(b_{\frac{n}{2}+1} - c_{\frac{n}{2}+1}\, g_{\frac{n}{2}+2})(d_{\frac{n}{2}} - a_{\frac{n}{2}}\, h_{\frac{n}{2}-1,0}) - c_{\frac{n}{2}}(d_{\frac{n}{2}+1} - c_{\frac{n}{2}+1}\, h_{\frac{n}{2}+2,0})} \tag{13}$$

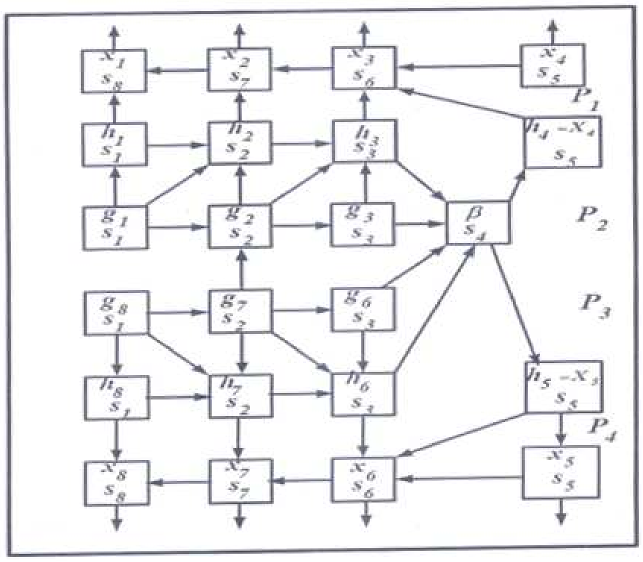

Figure 3: Parallel Splitting Gauss Jordan for *n* = 8

For Example *n = 8*, then the computations proceeds as follows:

<u>for sequential level 1 (*i = 1*)</u>

Compute in parallel, g_1, $h_{1,0}$, and g_8, $h_{8,0}$ using the parallel Gauss-Jordan algorithm 4 and the backward Gauss-Jordan algorithm 12 respectively.

<u>for sequential level 2 (*i = 2*)</u>

Compute in parallel, g_2, $h_{2,0}$ and g_7, $h_{7,0}$ using the parallel Gauss-Jordan algorithm 4 and the backward Gauss-Jordan algorithm 12 respectively, and also in parallel compute

$h_{1,1} = h_{1,0} - g_1 h_{2,0}$ using algorithm 4.

$h_{8,1} = h_{8,0} - g_8 h_{7,0}$ using algorithm 12.

<u>for sequential level 3 (i = 3)</u>

Compute in parallel, g_3, $h_{3,0}$ and g_6, $h_{6,0}$ using the parallel Gauss-Jordan algorithm 4 and the backward Gauss-Jordan algorithm 12 respectively, and also in parallel compute

$h_{2,1} = h_{2,0} - g_2 h_{3,0}$ using algorithm 4.

$h_{1,2} = h_{1,1} + g_1 g_2 h_{3,0}$ using algorithm 4.

$h_{7,1} = h_{7,0} - g_7 h_{6,0}$ using algorithm 12.

$h_{8,2} = h_{8,1} + g_7 g_8 h_{6,0}$using algorithm 12.

<u>for sequential level 4 (i – 4)</u>

Compute β from g_3, g_6, $h_{3,0}$ and $h_{6,0}$ from equation 13

<u>for sequential level 5 (i = 5)</u>

In parallel compute $x_4 = h_{4,0}$, and $x_5 = h_{5,0}$ using the parallel Gauss-Jordan algorithm 4 and the backward Gauss-Jordan algorithm 12 respectively.

<u>for sequential level 6 (i =6)</u>

Compute in parallel

$h_{3,1} = h_{3,0} - g_3 h_{4,0}$ *using algorithm 4.*

$h_{2,2} = h_{2,1} + g_2 g_3 h_{4,0}$ *using algorithm 4.*

$h_{1,3} = h_{1,2} - g_1 g_2 g_3 h_{4,0}$ *using algorithm 4.*

$h_{6,1} = h_{6,0} + g_6 g_{6,0}$ *using algorithm 12.*

$h_{7,2} = h_{7,1} - g_6 g_7 h_{5,0}$ *using algorithm 12.*

$h_{8,3} = h_{8,2} + g_6 g_7 g_8 h_{5,0}$ *using algorithm 12.*

and the solution can be obtained immediately in parallel at this step such that $x_i = h_{i,\frac{n}{2}-i,} \quad i = 1,2 \ldots \frac{n}{2}$ using algorithm 4 and such that

$$x_i = h_{i,i-\left(\frac{n}{2}+1\right),} \quad i = n, n-1, \ldots, \frac{n}{2}+1$$ using algorithm 12, see fig 4.

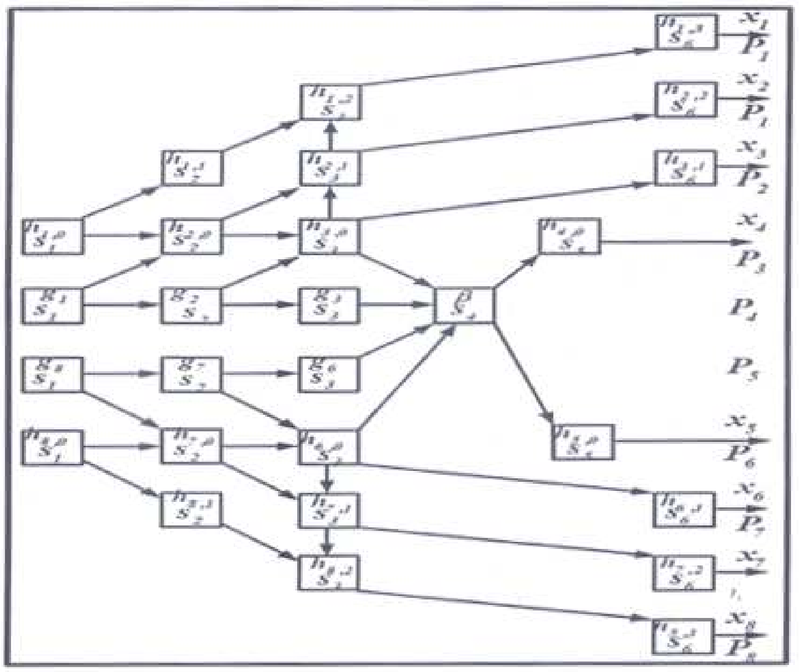

Figure 4: General Parallel Splitting Gauss Jordan for n = 8

6. COMMENTS

1. In this paper we consider a general tridiagonal system of equations with order *n*, and in the previous two sections we considered *n* to be even, now we consider *n* to be odd, in this case, the only change in the parallel splitting Gauss-Jordan algorithm and in the general parallel splitting Gauss-Jordan algorithm that we split the systems after the $\frac{n-1}{2}$ and before the $\frac{n+1}{2}$ equations to get 2 non equal systems of equations. In this case β in the parallel splitting Gauss-Jordan algorithm can be obtained as:

$$\beta = \frac{(b_{\frac{n+1}{2}} - a_{\frac{n+1}{2}} g_{\frac{n-1}{2}})(d_{\frac{n+3}{2}} - c_{\frac{n+3}{2}} h_{\frac{n+5}{2}}) - a_{\frac{n+3}{2}}(d_{\frac{n+1}{2}} - a_{\frac{n+1}{2}} h_{\frac{n-1}{2}})}{(b_{\frac{n+3}{2}} - c_{\frac{n+3}{2}} g_{\frac{n+5}{2}})(d_{\frac{n+1}{2}} - a_{\frac{n+1}{2}} h_{\frac{n-1}{2}}) - c_{\frac{n+1}{2}}(d_{\frac{n+3}{2}} - c_{\frac{n+3}{2}} h_{\frac{n+5}{2}})} \tag{14}$$

and β in the general parallel splitting Gauss-Jordan algorithm can be obtained as:

$$\beta = \frac{(b_{\frac{n+1}{2}} - a_{\frac{n+1}{2}} g_{\frac{n-1}{2}})(d_{\frac{n+3}{2}} - c_{\frac{n+3}{2}} h_{\frac{n+5}{2},0}) - a_{\frac{n+3}{2}}(d_{\frac{n+1}{2}} - a_{\frac{n+1}{2}} h_{\frac{n-1}{2},0})}{(b_{\frac{n+3}{2}} - c_{\frac{n+3}{2}} g_{\frac{n+5}{2}})(d_{\frac{n+1}{2}} - a_{\frac{n+1}{2}} h_{\frac{n-1}{2},0}) - c_{\frac{n+1}{2}}(d_{\frac{n+3}{2}} - c_{\frac{n+3}{2}} h_{\frac{n+5}{2},0})} \tag{15}$$

2. For the parallel splitting Gauss-Jordan algorithm if $x_{\frac{n}{2}+1} = 0$ which implies that $\beta = 0$ in this case

$$x_{\frac{n}{2}} = h_{\frac{n}{2}} = \frac{d_{\frac{n}{2}} - a_{\frac{n}{2}} h_{\frac{n}{2}-1}}{b_{\frac{n}{2}} - a_{\frac{n}{2}} g_{\frac{n}{2}-1}} \tag{16}$$

and $x_{\frac{n}{2}+2} \equiv h_{\frac{n}{2}+2}$

also for the general parallel splitting Gauss-Jordan algorithm if $x_{\frac{n}{2}+1} = 0$ which implies that β = 0 in this case:

$$x_{\frac{n}{2}} = h_{\frac{n}{2},0} = \frac{d_{\frac{n}{2}} - a_{\frac{n}{2}} h_{\frac{n}{2}-1,0}}{b_{\frac{n}{2}} - a_{\frac{n}{2}} g_{\frac{n}{2}-1}} \tag{17}$$

and $x_{\frac{n}{2}+2} \equiv h_{\frac{n}{2}+2,0}$

3. For the parallel splitting Gauss-Jordan algorithm if $x_{\frac{n}{2}} = 0$ which implies that

β is undefended in this case: $$x_{\frac{n}{2}+1} = h_{\frac{n}{2}+1} = \frac{d_{\frac{n}{2}+1} - c_{\frac{n}{2}+1} h_{\frac{n}{2}+2}}{b_{\frac{n}{2}+1} - c_{\frac{n}{2}+} g_{\frac{n}{2}+2}} \tag{18}$$

and $x_{\frac{n}{2}-1} \equiv h_{\frac{n}{2}-1}$

also for the general parallel splitting Gauss-Jordan algorithm with $x_{\frac{n}{2}} = 0$ which implies that β is undefinded in this case:

$$x_{\frac{n}{2}+1} = h_{\frac{n}{2}+1,0} = \frac{d_{\frac{n}{2}+1} - c_{\frac{n}{2}+1} h_{\frac{n}{2}+2,0}}{b_{\frac{n}{2}+1} - c_{\frac{n}{2}+} g_{\frac{n}{2}+2}} \tag{19}$$

and $x_{\frac{n}{2}-1} \equiv h_{\frac{n}{2}-1,0}$

For the parallel splitting Gauss-Jordan algorithm if $x_{\frac{n}{2}} = x_{\frac{n}{2}+1} = 0$

which implies that $\beta = \frac{0}{0}$ (undefined) then $x_{\frac{n}{2}-1} \equiv h_{\frac{n}{2}-1}$, $x_{\frac{n}{2}-2} \equiv h_{\frac{n}{2}+2}$

and for general parallel splitting Gauss-Jordan algorithm if

$x_{\frac{n}{2}} = x_{\frac{n}{2}+1} = 0$

which implies that $\beta = \frac{0}{0}$ (undefined) then:
$$x_{\frac{n}{2}-1} \equiv h_{\frac{n}{2}-1,0}$$
$$x_{\frac{n}{2}+2} \equiv h_{\frac{n}{2}+2,0}$$

7. RESULTS AND PERFORMANCE

7.1. Results

In this section we will examine some test problems on using the parallel splitting Gauss-Jordan algorithm and the general parallel splitting algorithm to deal with different types of tridiagonal systems of equations

1. .**Example 1: on using the parallel splitting Gauss-Jordan algorithm:** Consider the tridiagonal system of equations with n = 8 (n is even)

$$\begin{pmatrix} 4 & 1 & & & & & & \\ -1 & 2 & 2 & & & & & \\ & -3 & 5 & 4 & & & & \\ & & 4 & -2 & 5 & & & \\ & & & 3 & -1 & 4 & & \\ & & & & -3 & 5 & 2 & \\ & & & & & -7 & 4 & 1 \\ & & & & & & -3 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 6 \\ 9 \\ 25 \\ 29 \\ 31 \\ -6 \\ -5 \end{pmatrix} \qquad (20)$$

With exact solution $x_i = i, \quad i = 1, 2, \ldots, 8$

- **Computational Steps**

1. In parallel we get $g_1 = 0.25, g_8 = -1.5$, $h_1 = 1.5$ and $h_8 = -2.5$
2. In parallel we get $g_2 = 0.8888889$, $g_7 = -1.272727$, $h_2 = 4.666667$ and $h_7 = -0.636364$
3. In parallel we get $g_3 = 0.521739$, $g_6 = -0.39759$, $h_3 = 5.086956$ and $h_6 = 4.012049$
4. We get β = 1.25.
5. In parallel we get $x_4 = h_4 = 4$, $x_5 = h_5 = 5$.
6. In parallel we get $x_3 = h_3 = 3$, $x_6 = h_6 = 6$
7. In parallel we get $x_2 = h_2 = 2$, $x_7 = h_7 = 7$.
8. In parallel we get $x_1 = h_1 = 1$, $x_8 = h_8 = 8$.

2. **Solving example 1 using the general parallel splitting Gauss-Jordan algorithm:**

- **Computational Steps**

1. In parallel we get $g_1=0.25$, $g_8=-1.5$, $h_{1,0} =1.5$ and $h_{8,0} = -2.5$
2. In parallel we get $g2 = 0.8888889$, $g7 = -1.272727$, $h2,0 = 4.666667$, $h_{1,1} = .333333$, $h_{7,0} = -0.636364$, $h_{8,1} = -3.454546$
3. In parallel we get $g_3 = 0.521739$, $g_6 = -0.39759$, $h_{3,0} = 5.086956$ $h_{6,0} = 4.012049$, $h_{2,1} = .144928$, $h_{1,2} = 1.463768$,

$h_{7,1} = 4.469879$ *and* $h_{8,2} = 4.204819$

4. We get β = 1.25.

5. In parallel we get $x_4 = h_{4,0} = 4$, $x_5 = h_{5,0} = 5$,

6. In parallel we get $x_1 = h_{1,3} = 1$, $x_2 = h_{2,2} = 2$, $x_3 = h_{3,1} = 3$, $x_6 = h_{6,1} = 6$, $x_7 = h_{7,2} = 7$, $x_8 = h_{8,3} = 8$

3. **Example 2: It is example 1 (with x4 = 0),** so it has the same matrix of coefficients a with the right hand side vector $(6, 9, 9, 37, 19, 29, –6, –5)^T$, computational steps:

1. In parallel we get $g_1 = 0.25$, $g_8 = –1.5$, $h_1 = 1.5$ and $h_8 = –2.5$
2. In parallel we get $g_2 = 0.8888889$, $g_7 = –1.272727$, h_2=4.666667 and $h_7 = –0.636364$
3. In parallel we get $g_3 = 0.521739$, $g_6 = –0.39759$, h_3=3 and h_6= 4.012049
4. We get β = undefined and hence $x_4 = 0$
5. In parallel, $x_5 = h_5 = 5$ using equation 18
6. In parallel we get $x_3 = h_3 = 3$, $x_6 = h_6 = 6$
7. In parallel we get $x_2 = h_2 = 2$, $x_7 = h_7 = 7$
8. In parallel we get $x_1 = h_1 = 1$, $x_8 = h_8 = 8$

4. **Example 2: It is example 1 (with x5 = 0)** so it has the same matrix of coefficients A with the right hand side vector $(6, 9, 9, 12, 24, 44, –6, –5)^T$, and using the parallel splitting Gauss-Jordan algorithm Computational steps:

1. In parallel we get $g_1 = 0.25$, $g_8 = –1.5$, $h_1 = 1.5$ and $h_8 = –2.5$
2. In parallel we get $g_2 = 0.8888889$, $g_7 = –1.272727$, h_2=4.666667 and $h_7 = –0.636364$
3. In parallel we get $g_3 = 0.521739$, $g_6 = –0.39759$, h_3=5.086956 and $h_6 = 6$
4. We get β = 0 and hence $x_5 = 0$
5. In parallel, $x_4 = h_4 = 4$ using equation 16.
6. In parallel we get $x_3 = h_3 = 3$, $x_6 = h_6 = 6$
7. In parallel we get $x_2 = h_2 = 2$, $x_7 = h_7 = 7$
8. In parallel we get $x_1 = h_1 = 1$, $x_8 = h_8 = 8$

7.2. Performance

The two new parallel algorithms: Parallel splitting Gauss – Jordan and the general parallel splitting Gauss Jordan algorithms are very easily distributed over different number of processors Also our implementations are very efficient, generally a linear speed-up obtained by increasing the number of processors. In addition, we do not any restrictions on the shape of the tridiagonal system of equation, or on n. In general for $n \times n$ tridiagonal system, the parallel Gauss-Jordan algorithm, needs n sequential steps using $n + 1$ processors, and has the advantage, that all solutions can be obtained immediately at the same time (in science, you can deal with *n* objects at the same time). The parallel splitting Gauss-Jordan algorithm needs *n* sequential steps, and has the advantage that it uses only $\frac{n}{2}$ processors, but solutions can be obtained in parallel pairs.

The general parallel splitting Gauss-Jordan algorithm has the advantages that it only needs $\frac{n}{2}+2$ sequential steps using *n* processors, and also, all solutions can be obtained immediately at the same time. Also these Parallel algorithms can easily be extended to solve the block tridiagonal system of equations see [1].

8. CONCLUSIONS

The two new parallel algorithms: Parallel splitting Gauss – Jordan and the general parallel splitting Gauss Jordan algorithms are very easily distributed over different number of processors Also our implementations are very efficient, generally a linear speed-up obtained by increasing the number of processors. In addition, we do not any restrictions on the shape of the tridiagonal system of equation, or on *n*. In general for $n \times n$ tridiagonal system, the parallel Gauss-Jordan algorithm, needs *n* sequential steps using $n + 1$ processors, and has the advantage, that all solutions can be obtained immediately at the same time (in science, you can deal with *n* objects at the same time). The parallel splitting Gauss-Jordan algorithm needs *n* sequential steps, and has the advantage that it uses only $\frac{n}{2}$ processors, but solutions can be obtained in parallel pairs.

The general parallel splitting Gauss-Jordan algorithm has the advantages that it only needs $\frac{n}{2}+2$ sequential steps using *n* processors, and also, all solutions can be obtained immediately at the same time. Also these Parallel algorithms can easily be extended to solve the block tridiagonal system of equations see [1].

REFERENCES

1. J. Evans., "New parallel Algorithms for Partial Differential Equations," *M. Feilmeier, J. Joubert. and U. Schendel, editors, Int. Conf. Parallel Computing 83,* pages 52-55. – 1984.
2. M. Hatzopoulos "Parallel linear system solvers for tridiagonal systems." *J. F. Traub., editors, Complexity of Sequential and Parallel Numerical Algorithms.*, pages 387-393, - 1979.
3. Henk. A. vn der Vorst "Analysis of Parallel solution method for tridiagonal systems.", *Reports of the Department of Mathematics, Delft,* pages 86-96,- 1986.